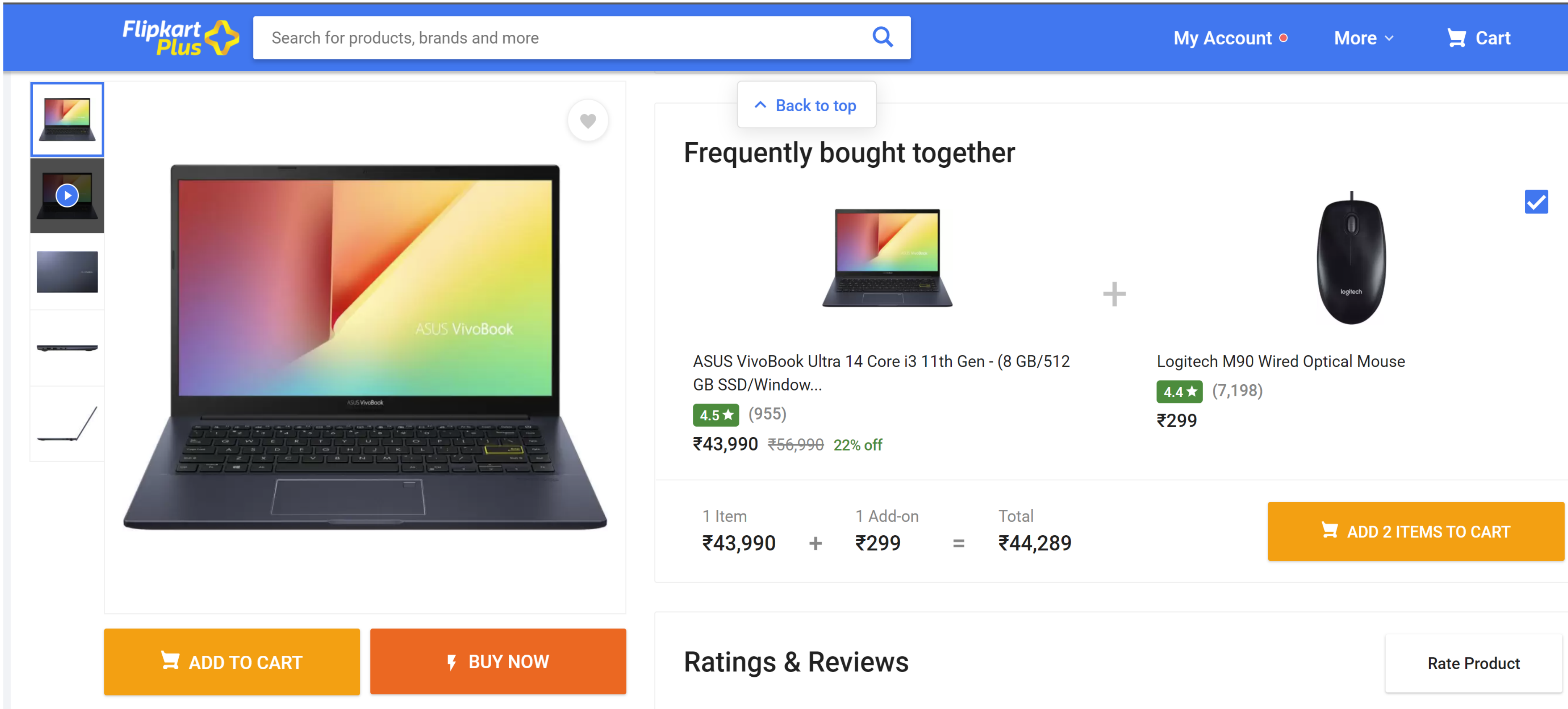
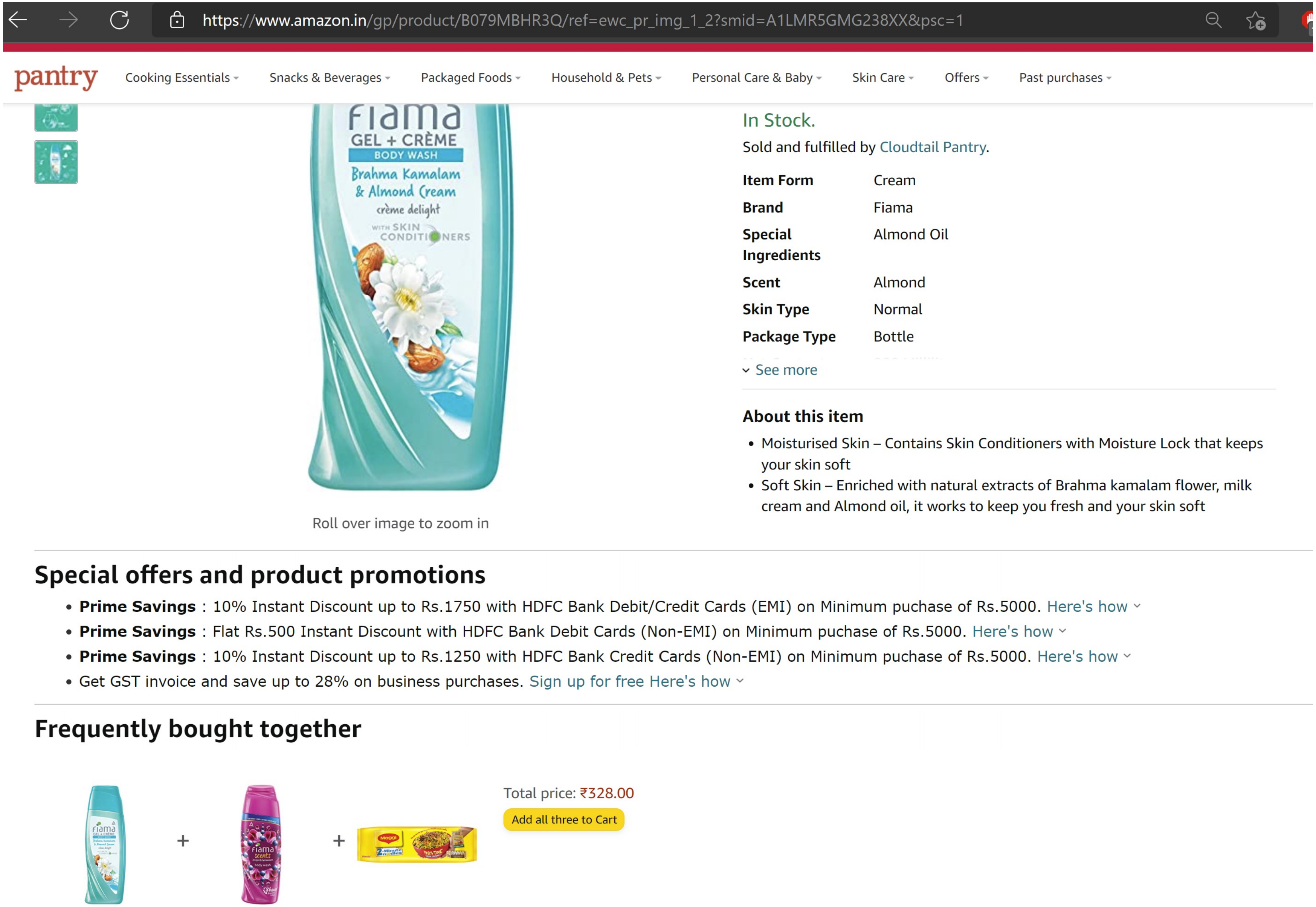
Association Rule Mining

Dr. Abhijnan Chakraborty
Department of Computer Science & Engg.,
Indian Institute of Technology Kharagpur
https://cse.iitkgp.ac.in/~abhijnan

Frequent Itemsets are Everywhere



Frequent Itemsets are Everywhere



Definition: Frequent Itemset

- Itemset
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- Support count (σ)
  - Frequency of occurrence of an itemset
  - E.g. σ({Milk, Bread,Diaper}) = 2
- Support
  - Fraction of transactions that contain an itemset
  - E.g. s({Milk, Bread, Diaper}) = 2/5
- Frequent Itemset
  - An itemset whose support is greater than or equal to a *minsup* threshold

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Butter, Eggs
3	Milk, Diaper, Butter, Coke
4	Bread, Milk, Diaper, Butter
5	Bread, Milk, Diaper, Coke

Association Rule Mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

TID	*Items*
1	**Bread, Milk**
2	**Bread, Diaper, Butter, Eggs**
3	**Milk, Diaper, Butter, Coke**
4	**Bread, Milk, Diaper, Butter**
5	**Bread, Milk, Diaper, Coke**

Example of Association Rules

$\{\text{Diaper}\} \rightarrow \{\text{Butter}\}$,
$\{\text{Milk, Bread}\} \rightarrow \{\text{Eggs, Coke}\}$,
$\{\text{Butter, Bread}\} \rightarrow \{\text{Milk}\}$,

Implication means co-occurrence, not causality!

Definition: Association Rule

- Association Rule
  - An implication expression of the form $X \to Y$, where X and Y are itemsets
  - Example:
    $\{\text{Milk, Diaper}\} \to \{\text{Butter}\}$

TID	*Items*
1	**Bread, Milk**
2	**Bread, Diaper, Butter, Eggs**
3	**Milk, Diaper, Butter, Coke**
4	**Bread, Milk, Diaper, Butter**
5	**Bread, Milk, Diaper, Coke**

- Rule Evaluation Metrics
  - Support (s)
    - Fraction of transactions that contain both X and Y
  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

Example:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \{\text{Butter}\}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Butter})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Butter})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ *minsup* threshold
  - confidence ≥ *minconf* threshold

- **Brute-force approach:**
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds

  ⇒ Computationally prohibitive!

Mining Association Rules

TID	*Items*
1	**Bread, Milk**
2	**Bread, Diaper, Butter, Eggs**
3	**Milk, Diaper, Butter, Coke**
4	**Bread, Milk, Diaper, Butter**
5	**Bread, Milk, Diaper, Coke**

Example of Rules:

{Milk,Diaper} $\rightarrow$ {Butter} (s=0.4, c=0.67)
{Milk,Butter} $\rightarrow$ {Diaper} (s=0.4, c=1.0)
{Diaper,Butter} $\rightarrow$ {Milk} (s=0.4, c=0.67)
{Butter} $\rightarrow$ {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} $\rightarrow$ {Milk,Butter} (s=0.4, c=0.5)
{Milk} $\rightarrow$ {Diaper,Butter} (s=0.4, c=0.5)

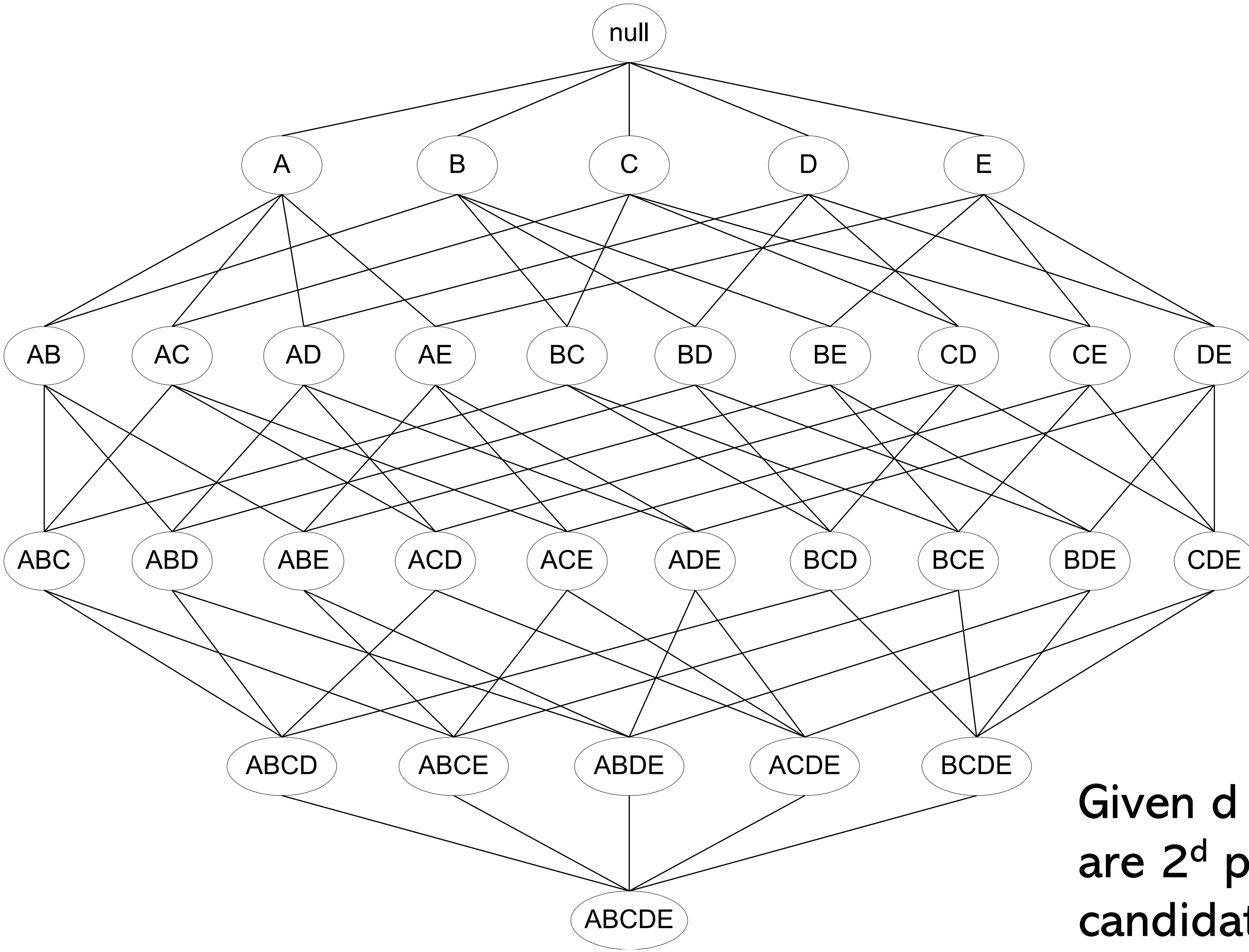
Observations:

- All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Butter}
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements

Mining Association Rules

- Two-step approach:
  1. **Frequent Itemset Generation**
     - Generate all itemsets whose support $\geq$ minsup
  2. **Rule Generation**
     - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
- Frequent itemset generation is still computationally expensive

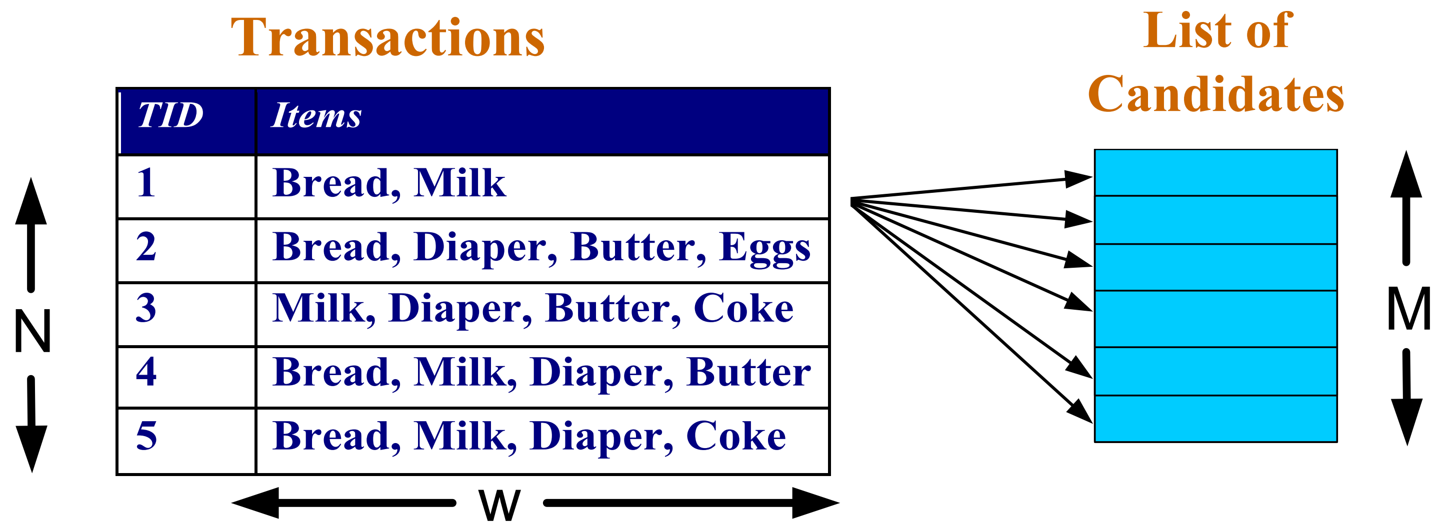
Frequent Itemset Generation



Given d items, there are 2^d possible candidate itemsets

Frequent Itemset Generation

- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database



TID	*Items*
1	**Bread, Milk**
2	**Bread, Diaper, Butter, Eggs**
3	**Milk, Diaper, Butter, Coke**
4	**Bread, Milk, Diaper, Butter**
5	**Bread, Milk, Diaper, Coke**

  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => Expensive since $M = 2^d$!!!

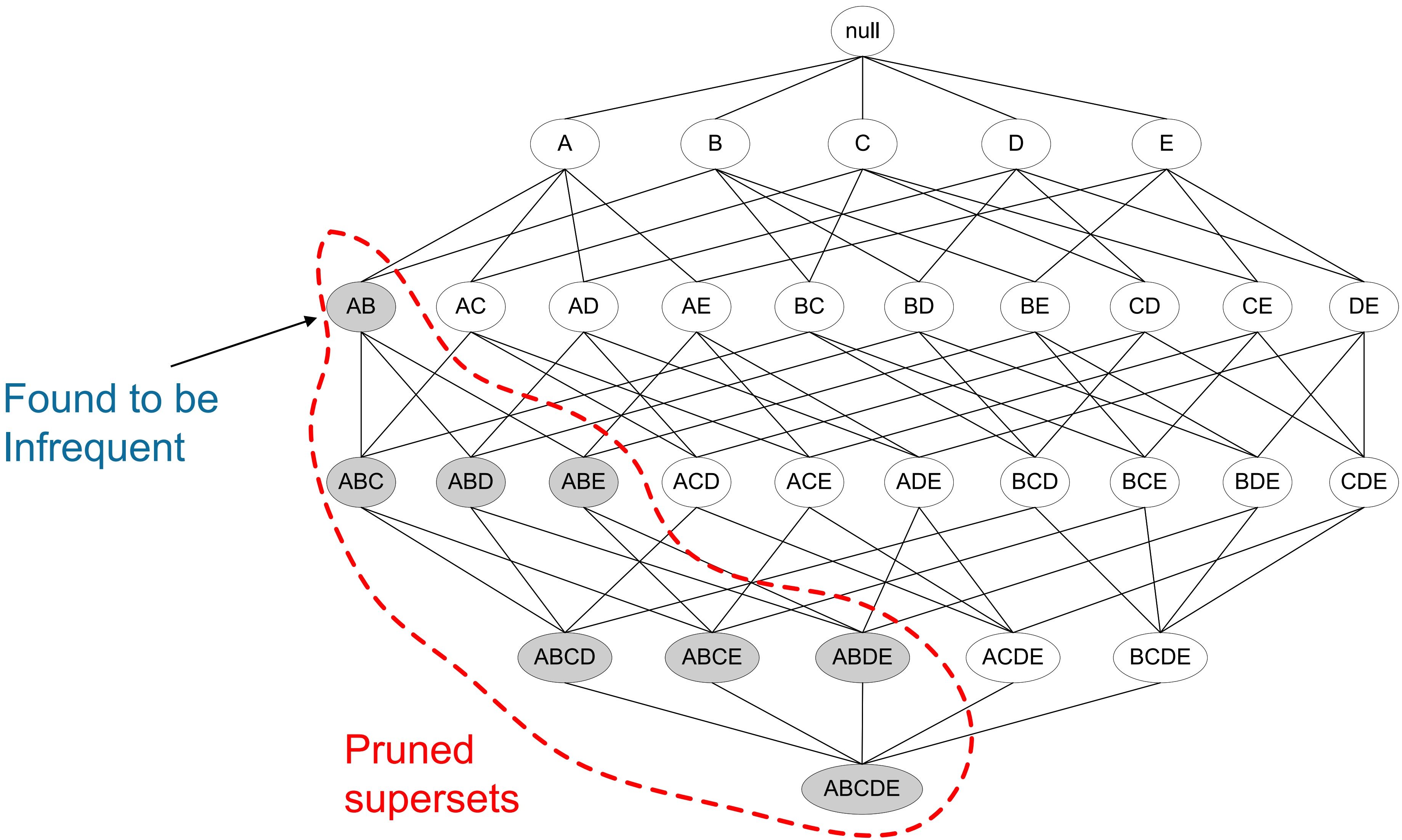
Reducing Number of Candidates

- Apriori principle:
  - If an itemset is frequent, then all its subsets must also be frequent
- Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

  - Support of an itemset never exceeds the support of its subsets
  - This is known as the anti-monotone property of support

Illustrating Apriori Principle



Illustrating Apriori Principle

TID	*Items*
1	Bread, Milk
2	Butter, Bread, Diaper, Eggs
3	Butter, Coke, Diaper, Milk
4	Butter, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Butter	**3**
Diaper	**4**
Eggs	**1**

If every subset is considered,

$^6C_1 + {}^6C_2 + {}^6C_3$

$6 + 15 + 20 = 41$

Illustrating Apriori Principle

TID	*Items*
1	Bread, Milk
2	Butter, Bread, Diaper, Eggs
3	Butter, Coke, Diaper, Milk
4	Butter, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

Items (1-itemsets)

Item	Count
Bread	4
Coke	2
Milk	4
Butter	3
Diaper	4
Eggs	1

Minimum Support = 3

If every subset is considered,

$^6C_1 + {}^6C_2 + {}^6C_3$

$6 + 15 + 20 = 41$

With support-based pruning,

$6 + 6 + 4 = 16$

Illustrating Apriori Principle

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Butter	**3**
Diaper	**4**
Eggs	**1**

Items (1-itemsets)

Itemset
{Bread,Milk}
{Bread, Butter }
{Bread,Diaper}
{Butter, Milk}
{Diaper, Milk}
{Butter,Diaper}

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

If every subset is considered,

$^6C_1 + {}^6C_2 + {}^6C_3$

$6 + 15 + 20 = 41$

With support-based pruning,

$6 + 6 + 4 = 16$

Illustrating Apriori Principle

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Butter	**3**
Diaper	**4**
Eggs	**1**

Itemset	Count
{Bread,Milk}	**3**
{Butter, Bread}	**2**
{Bread,Diaper}	**3**
{Butter,Milk}	**2**
{Diaper,Milk}	**3**
{Butter,Diaper}	**3**

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

If every subset is considered,

$^6C_1 + {}^6C_2 + {}^6C_3$

$6 + 15 + 20 = 41$

With support-based pruning,

$6 + 6 + 4 = 16$

Illustrating Apriori Principle

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Butter	**3**
Diaper	**4**
Eggs	**1**

Pairs (2-itemsets)

Itemset	Count
{Bread,Milk}	**3**
{Bread,Butter}	**2**
{Bread,Diaper}	**3**
{Milk,Butter}	**2**
{Milk,Diaper}	**3**
{Butter,Diaper}	**3**

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

Triplets (3-itemsets)

Itemset
{ Butter, Diaper, Milk}
{ Butter,Bread,Diaper}
{Bread,Diaper,Milk}
{ Butter, Bread, Milk}

If every subset is considered,

$$^6C_1 + {}^6C_2 + {}^6C_3$$

$$6 + 15 + 20 = 41$$

With support-based pruning,

$$6 + 6 + 4 = 16$$

Illustrating Apriori Principle

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	2
Milk	**4**
Butter	**3**
Diaper	**4**
Eggs	1

Pairs (2-itemsets)

Itemset	Count
{Bread,Milk}	**3**
{Bread,Butter}	2
{Bread,Diaper}	**3**
{Milk,Butter}	2
{Milk,Diaper}	**3**
{Butter,Diaper}	**3**

(No need to generate candidates involving Coke or Eggs)

Triplets (3-itemsets)

Itemset	Count
{ Butter, Diaper, Milk}	2
{ Butter,Bread, Diaper}	2
{Bread, Diaper, Milk}	**2**
{Butter, Bread, Milk}	1

Minimum Support = 3

If every subset is considered,

$$^6C_1 + {}^6C_2 + {}^6C_3$$

$$6 + 15 + 20 = 41$$

With support-based pruning,

$$6 + 6 + 4 = 16$$

Illustrating Apriori Principle

TID	Items
1	Bread, Milk
2	Butter, Bread, Diaper, Eggs
3	Butter, Coke, Diaper, Milk
4	Butter, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Butter	**3**
Diaper	**4**
Eggs	**1**

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Itemset	Count
{Bread,Milk}	**3**
{Bread,Butter}	**2**
{Bread,Diaper}	**3**
{Milk,Butter}	**2**
{Milk,Diaper}	**3**
{Butter,Diaper}	**3**

Minimum Support = 3

Triplets (3-itemsets)

Itemset	Count
{ Butter, Diaper, Milk}	**2**
{ Butter,Bread, Diaper}	**2**
{Bread, Diaper, Milk}	**2**
{Butter, Bread, Milk}	**1**

If every subset is considered,

$^6C_1 + {}^6C_2 + {}^6C_3$

$6 + 15 + 20 = 41$

With support-based pruning,

$6 + 6 + 4 = 16$

$6 + 6 + 1 = 13$

Apriori Algorithm

- F_k: frequent k-itemsets; C_k: candidate k-itemsets
- Algorithm
  - Let k=1
  - Generate F_1 = {frequent 1-itemsets}
  - Repeat until F_k is empty
    - **Candidate Generation**: Generate C_{k+1} from F_k
    - **Candidate Pruning**: Prune candidate itemsets in C_{k+1} containing subsets of length k that are infrequent
    - **Support Counting**: Count the support of each candidate in C_{k+1} by scanning the transaction database
    - **Candidate Elimination**: Eliminate candidates in C_{k+1} that are infrequent, leaving only those that are frequent => F_{k+1}

Candidate Generation: F_{k-1} x F_{k-1} Method

- Introduction of ordering: items can be sorted in lexicographic order

- Merge two frequent (k-1)-itemsets if their first (k-2) items are identical

- F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE}
  - Merge(ABC, ABD) = ABCD
  - Merge(ABC, ABE) = ABCE
  - Merge(ABD, ABE) = ABDE

  - Do not merge(ABD,ACD) because they share only prefix of length 1 instead of length 2

Candidate Pruning

- Let F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE} be the set of frequent 3-itemsets

- C_4 = {ABCD,ABCE,ABDE} is the set of candidate 4-itemsets generated (from previous slide)

- Candidate pruning
  - Prune ABCE because ACE and BCE are infrequent
  - Prune ABDE because ADE is infrequent

- After candidate pruning: C_4 = {ABCD}

Illustrating Apriori Principle

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Butter	**3**
Diaper	**4**
Eggs	**1**

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Itemset	Count
{Bread,Milk}	**3**
{Bread,Butter}	**2**
{Bread,Diaper}	**3**
{Milk,Butter}	**2**
{Milk,Diaper}	**3**
{Butter,Diaper}	**3**

Minimum Support = 3

Triplets (3-itemsets)

Itemset	Count
{Bread, Diaper, Milk}	**2**

If every subset is considered,

$^6C_1 + {}^6C_2 + {}^6C_3$

$6 + 15 + 20 = 41$

With support-based pruning,

$6 + 6 + 1 = 13$

Use of $F_{k-1}xF_{k-1}$ method for candidate generation results in only one 3-itemset. This is eliminated after the support counting step.

Alternate $F_{k-1} \times F_{k-1}$ Method

- Merge two frequent (k-1)-itemsets if the last (k-2) items of the first one is identical to the first (k-2) items of the second.

- F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE}
  - Merge(A<u>BC</u>, <u>BC</u>D) = A<u>BC</u>D
  - Merge(A<u>BD</u>, <u>BD</u>E) = A<u>BD</u>E
  - Merge(A<u>CD</u>, <u>CD</u>E) = A<u>CD</u>E
  - Merge(B<u>CD</u>, <u>CD</u>E) = B<u>CD</u>E

Candidate Pruning for Alternate $F_{k-1} \times F_{k-1}$ Method

- Let F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE} be the set of frequent 3-itemsets

- C_4 = {ABCD,ABDE,ACDE,BCDE} is the set of candidate 4-itemsets generated (from previous slide)

- Candidate pruning
  - Prune ABDE because ADE is infrequent
  - Prune ACDE because ACE and ADE are infrequent
  - Prune BCDE because BCE

- After candidate pruning: C_4 = {ABCD}

Count Support of Candidate Itemsets

- Scan the database of transactions to determine the support of each candidate itemset

- Naïve counting:
  - For each candidate $I_i \in C_{k+1}$
    - For each transaction T_j in T
      - Check whether I_i appears in T_j

- This can be very slow if both $|C_{k+1}|$ and $|T|$ are large

Count Support with a Data Structure

- A Better Approach
  - Organize the candidate patterns in C_{k+1} in a data structure
- Use the data structure to accelerate counting
  - Each transaction in T_i examined against the subset of candidates in C_{k+1} that might be contained in T_i

Support Counting based on Hashing

Naïve counting:

For each $I_i \in C_{k+1}$
  For all $T_j \in T$
    If $I_i \subseteq T_j$
      Add to $sup(I_i)$

Hashed counting:

For each $T_j \in T$
  For $I_i \in hashbucket(T_j, C_{k+1})$
    If $I_i \subseteq T_j$
      Add to $sup(I_i)$

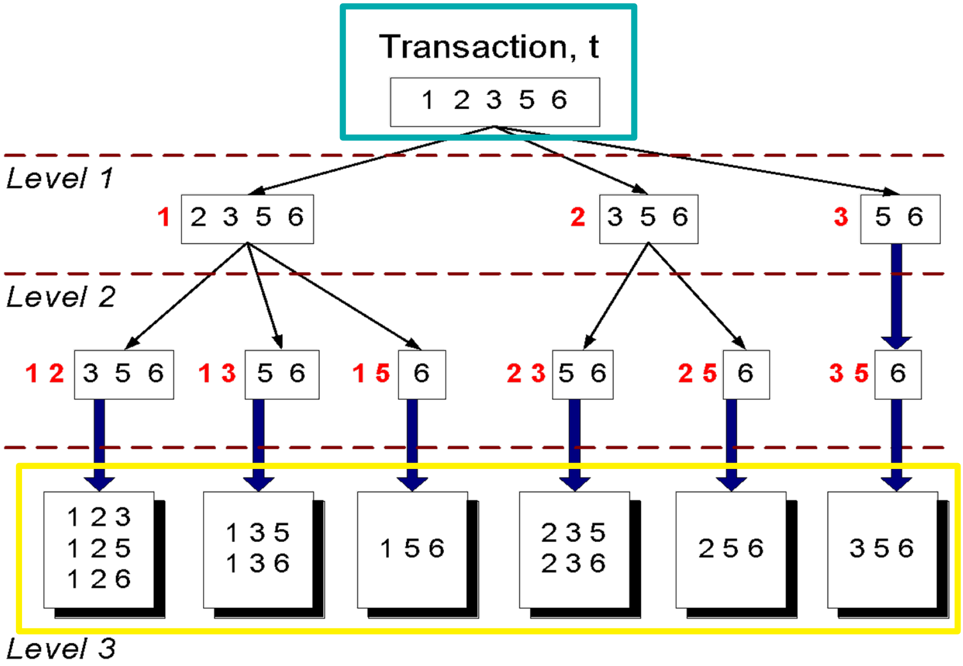
Which Candidates are Relevant?

Imagine 15 candidate itemsets of length 3:

{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

Now, suppose we look for this transaction:

{1 2 3 5 6}



Here we depict only the candidates that appear in the transaction (10 out of 15)

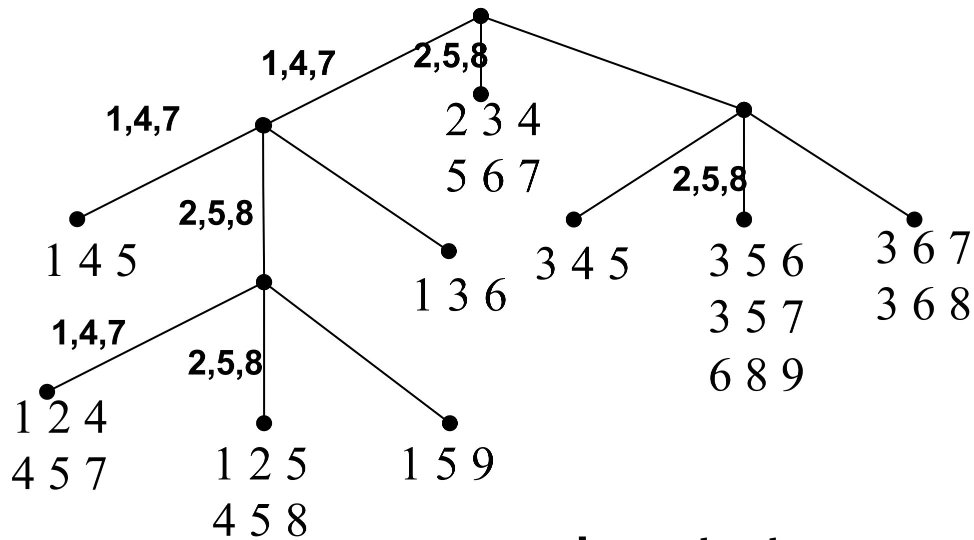
Hash Tree for Itemsets in C_{k+1}

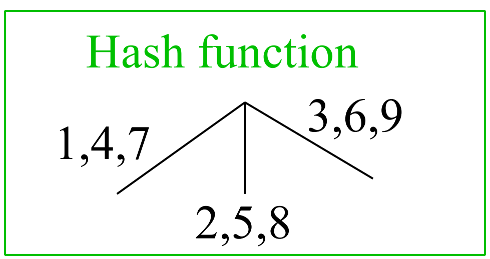
- A tree with fixed degree r
- Each itemset in C_{k+1} is stored in a leaf node
- All internal nodes use a hash function to map items to one of the r branches (can be the same for all internal nodes)
- All leaf nodes contain a lexicographically sorted list of up to max_leaf_size itemsets

Example Hash Tree

r=3 max_leaf_size=3

Candidate itemsets
{1 4 5}, {1 2 4}, {4 5 7},
{1 2 5}, {4 5 8}, {1 5 9},
{1 3 6}, {2 3 4}, {5 6 7},
{3 4 5}, {3 5 6}, {3 5 7},
{6 8 9}, {3 6 7}, {3 6 8}

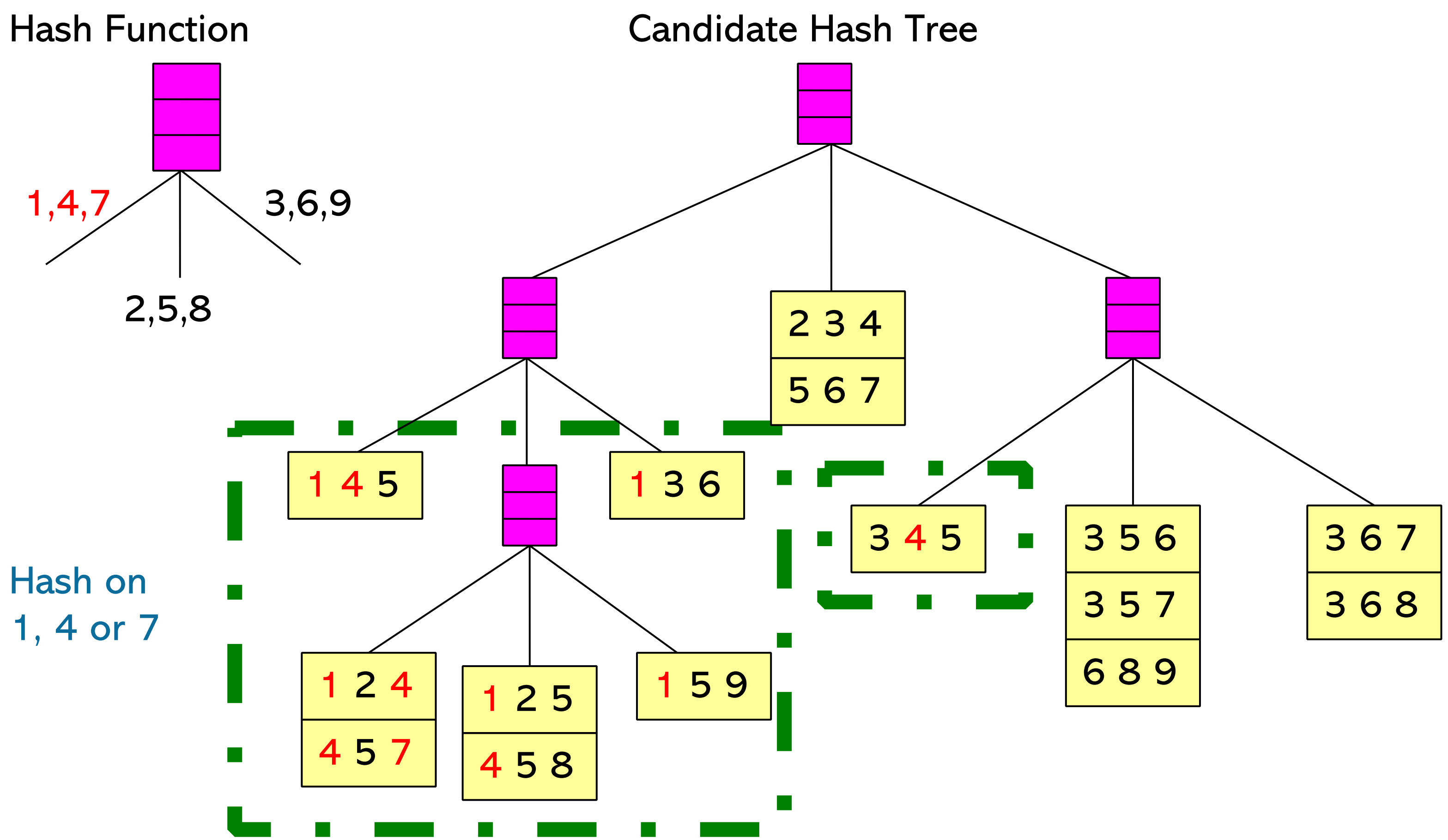




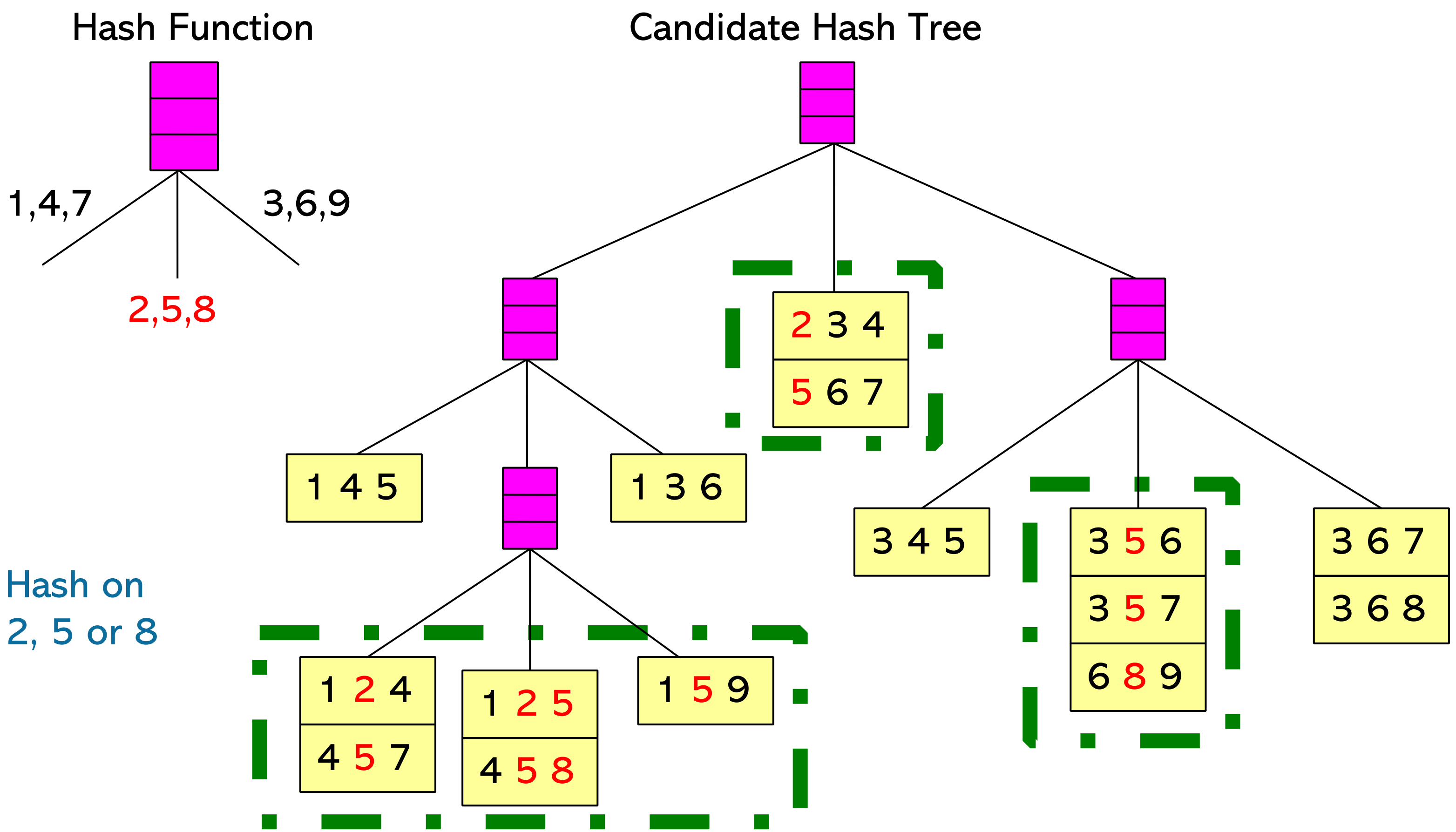
Important: itemsets are sorted!

$$h(p) = (p - 1) \bmod 3$$

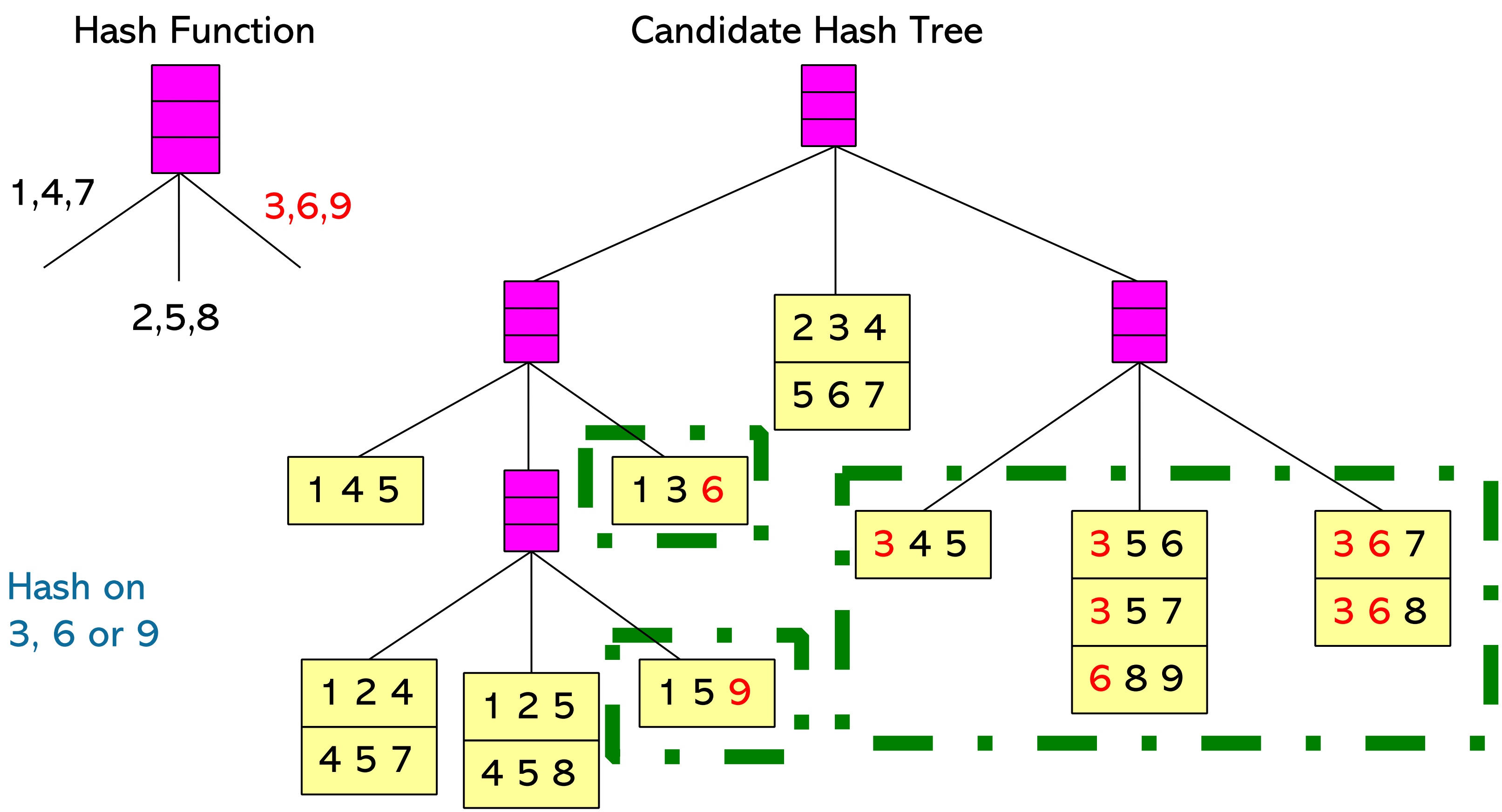
Example Hash Tree (Cont.)



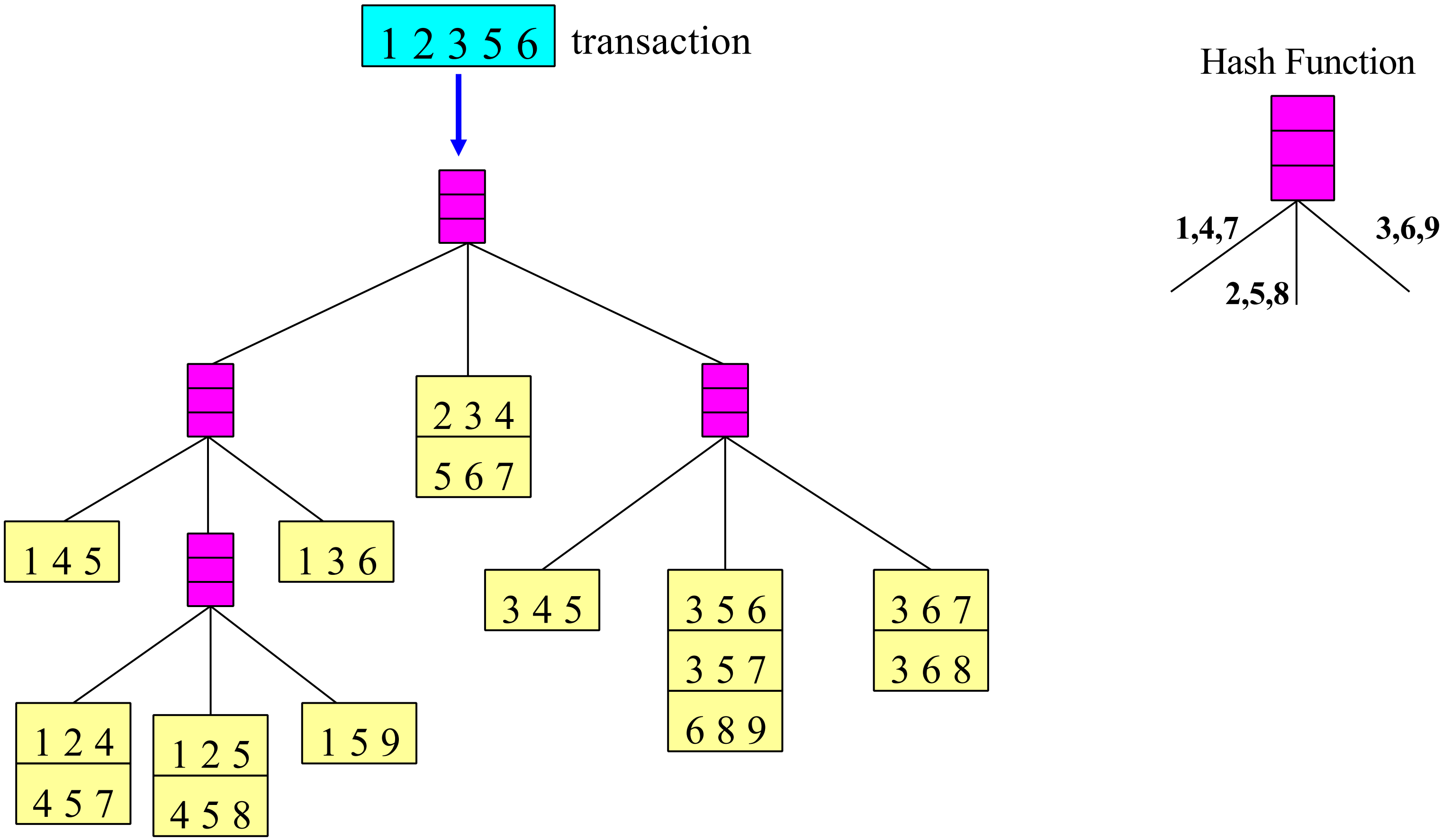
Example Hash Tree (Cont.)



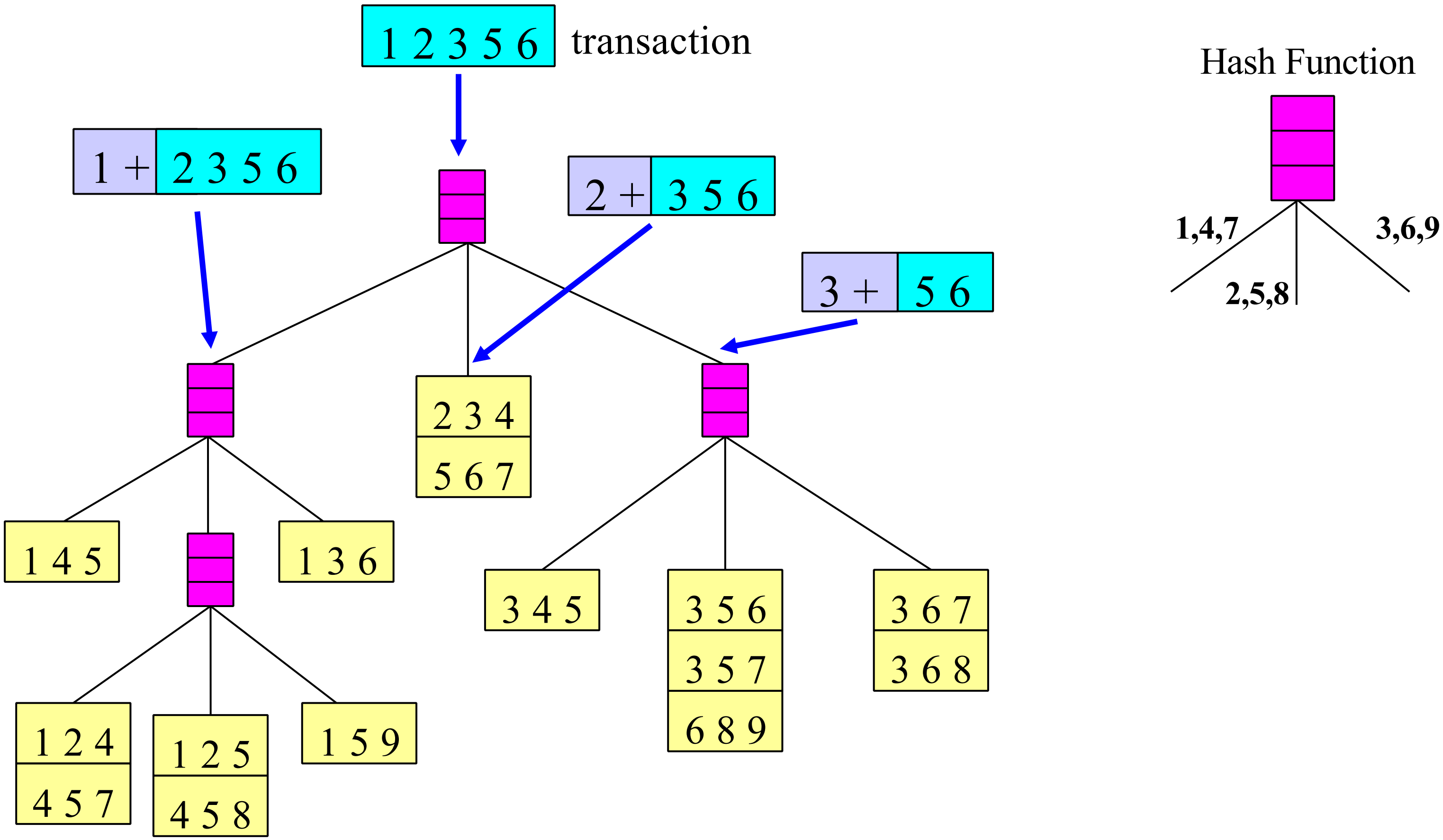
Example Hash Tree (Cont.)



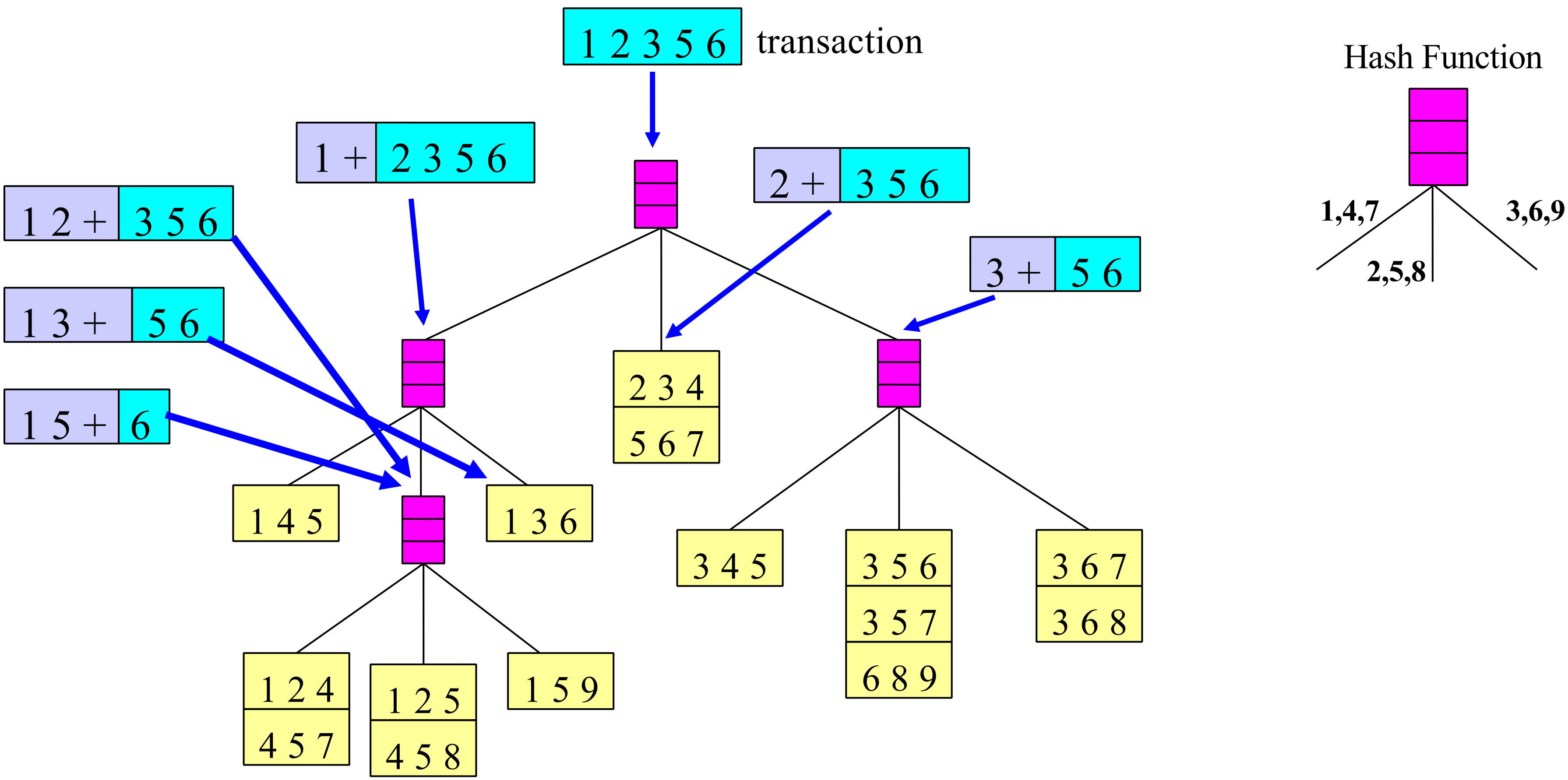
Checking which candidates might be in a transaction



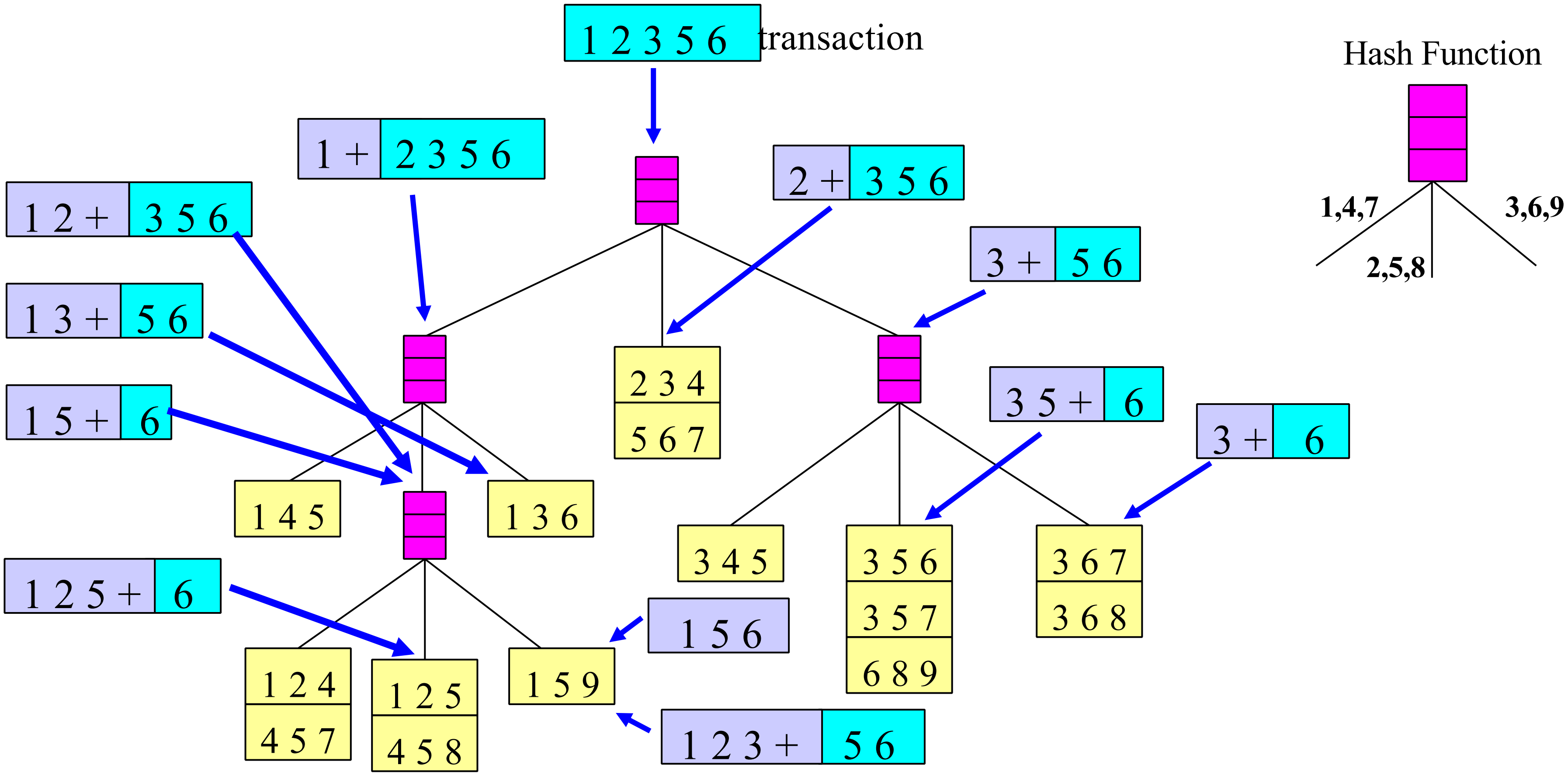
Checking which candidates might be in a transaction



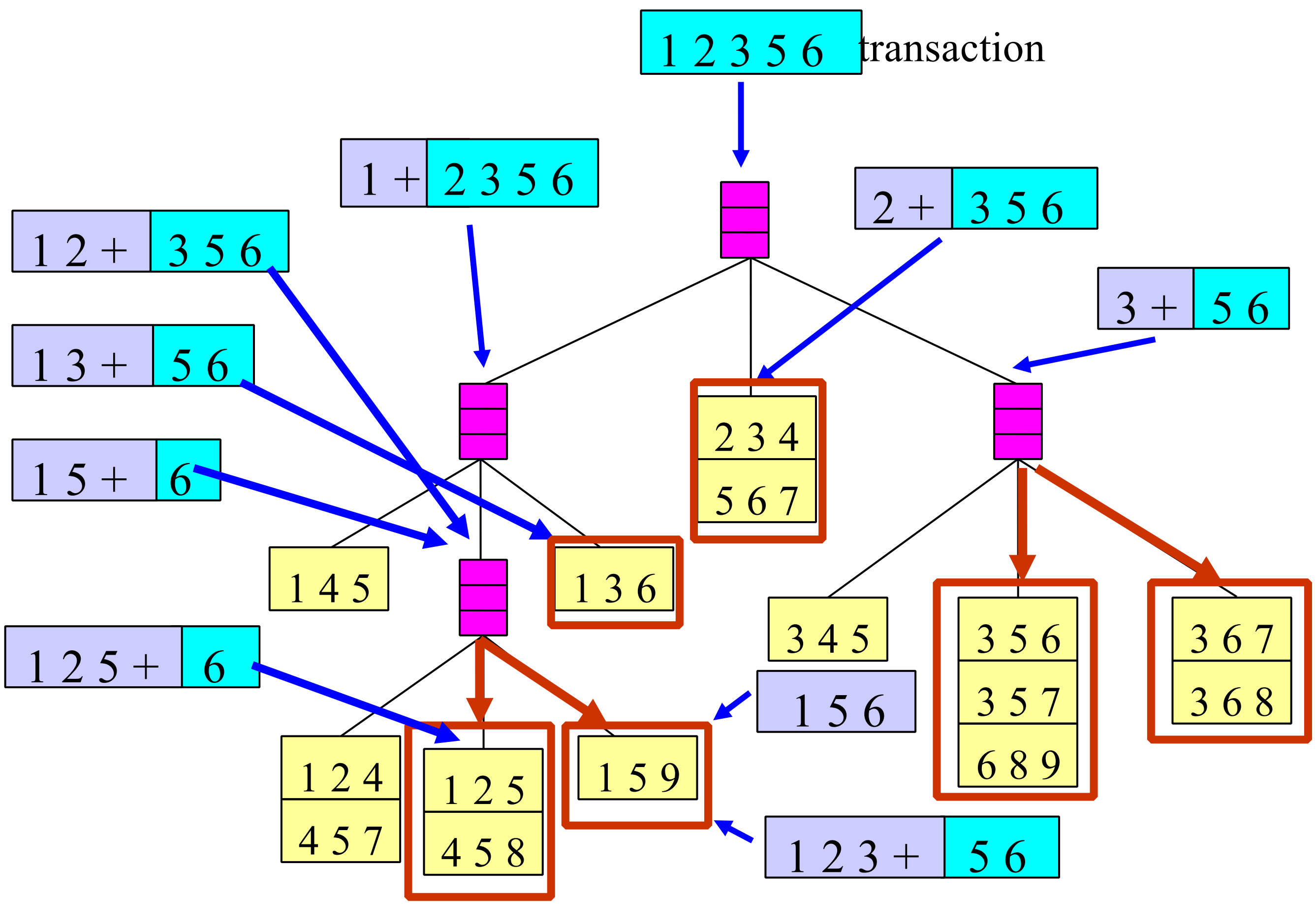
Checking which candidates might be in a transaction

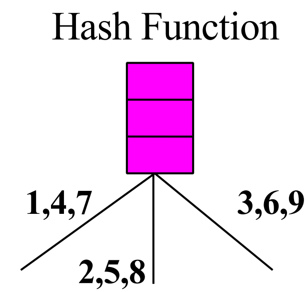


Checking which candidates might be in a transaction



Checking which candidates might be in a transaction





Compare transaction against 11 out of 15 candidates

Summary: Efficient Frequent Itemsets

- $C_1 \leftarrow$ singletons, lexicographically sorted
- $F_1 \leftarrow$ elements in C_1 with support $\geq$ minsup, obtained by direct counting
- $k \leftarrow 1$
- While F_k is not empty
  - Generate C_{k+1} by merging elements in F_k sharing a prefix of size k-1
  - Remove from C_{k+1} elements that do not have all of their subsets in F_k
  - Create hash tree for C_{k+1}
  - Pass all transactions in T by the hash tree to compute support for elements of C_{k+1}
  - $F_{k+1} \leftarrow$ elements in C_{k+1} with support $\geq$ minsup, lexicographically sorted
- Return the union of F_1, F_2, …, F_k

Rule Generation

- Given a frequent itemset L, find all non-empty subsets $f \subset L$ such that $f \to L - f$ satisfies the minimum confidence requirement
  - If {A,B,C,D} is a frequent itemset, candidate rules:

ABC →D,	ABD →C,	ACD →B,	BCD →A,
A →BCD,	B →ACD,	C →ABD,	D →ABC
AB →CD,	AC → BD,	AD → BC,	BC →AD,
BD →AC,	CD →AB,		

- If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $L \to \emptyset$ and $\emptyset \to L$)

Rule Generation

- In general, confidence does not have an anti-monotone property

  $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$

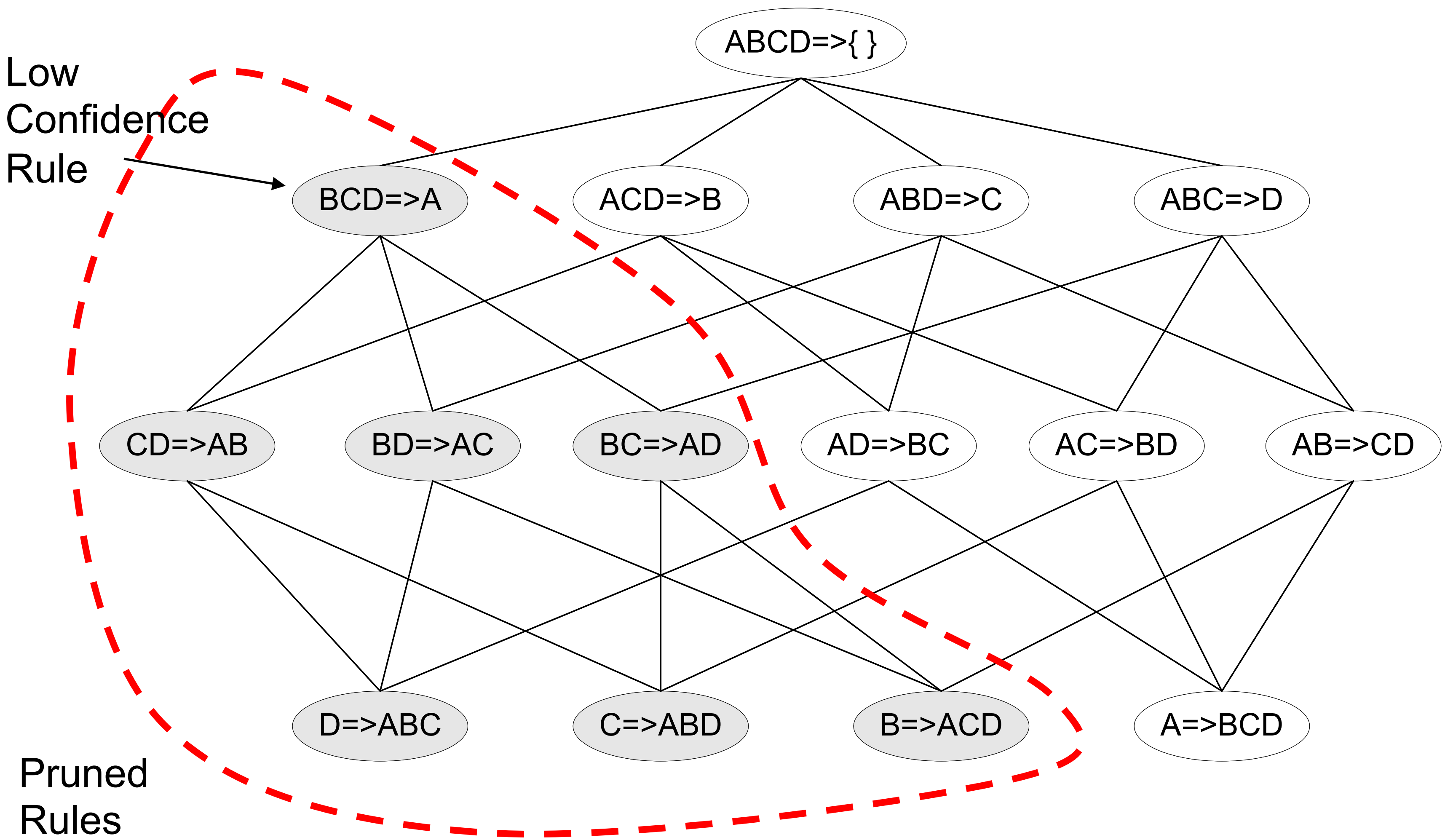
- But confidence of rules generated from the same itemset has an anti-monotone property
  - E.g., Suppose {A,B,C,D} is a frequent 4-itemset:

    $$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$

  - Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

Rule Generation for Apriori Algorithm

Lattice of rules



Exercise: Apriori

FInd all rules of the form

$\{a,b\} \longrightarrow \{c\}$

having:

support $\geq 2/9$ and

confidence $\geq 50\%$

Note: to generate only rules of the form $\{a,b\} \longrightarrow \{c\}$, consider only itemsets of size 3

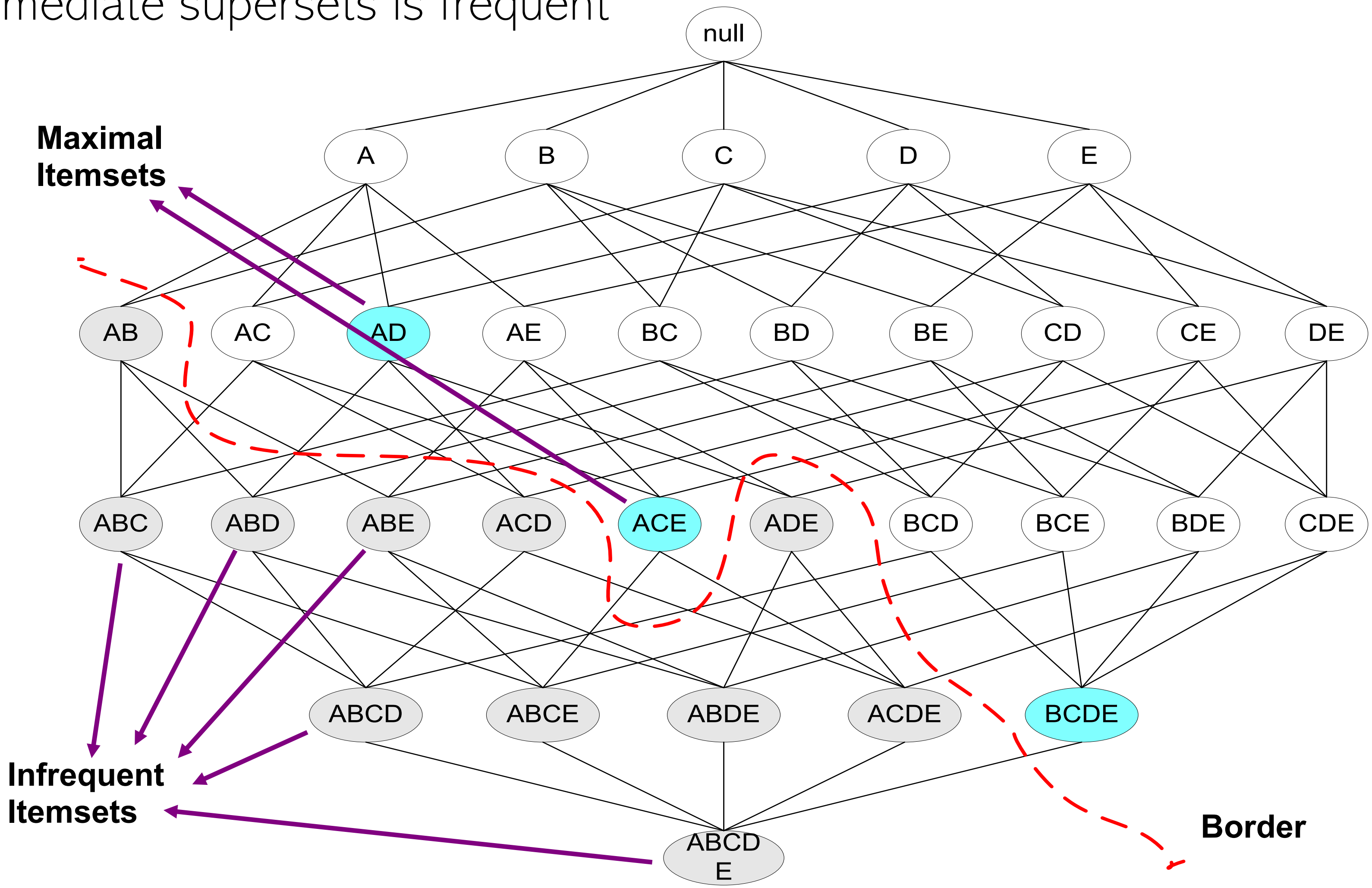
TID	items
T1	I1, I2 , I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

Compact Representation of Frequent Itemsets

- In practice, the number of frequent itemsets produced from a transaction data set can be very large
- It is useful to identify a small representative set of frequent itemsets from which all other frequent itemsets can be derived
- Two such representations are
  - Maximal frequent itemsets
  - Closed frequent itemsets

Maximal Frequent Itemset

An itemset is maximal frequent if it is frequent and none of its immediate supersets is frequent



Closed Itemset

- An itemset X is closed if none of its immediate supersets has the same support as the itemset X.
- X is not closed if at least one of its immediate supersets has support count as X.

Closed Itemset

- An itemset X is closed if none of its immediate supersets has the same support as the itemset X.
- X is not closed if at least one of its immediate supersets has support count as X.

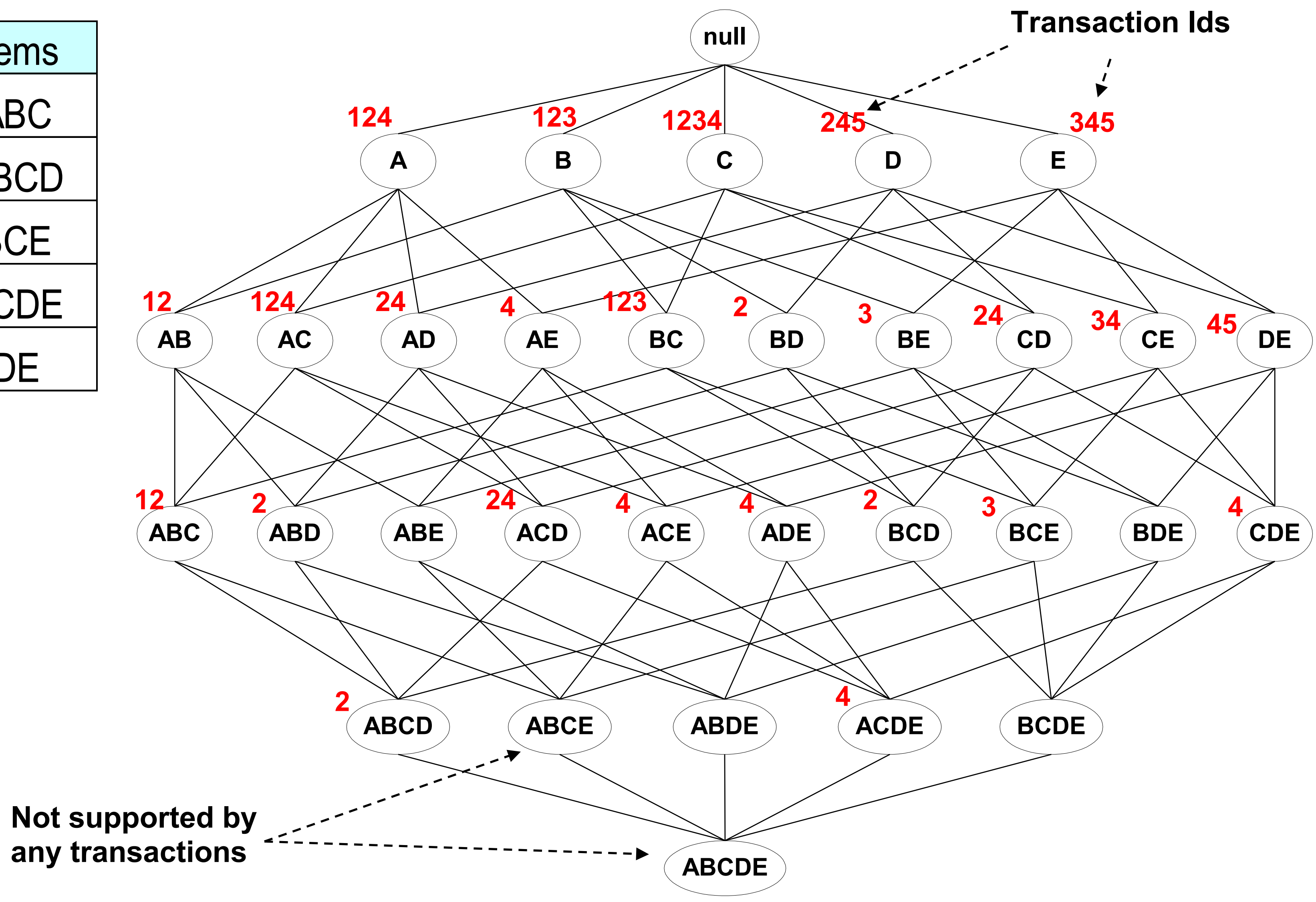
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

Itemset	Support
{A}	4
{B}	5
{C}	3
{D}	4
{A,B}	4
{A,C}	2
{A,D}	3
{B,C}	3
{B,D}	4
{C,D}	3

Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	2
{A,B,C,D}	2

Maximal vs Closed Itemsets

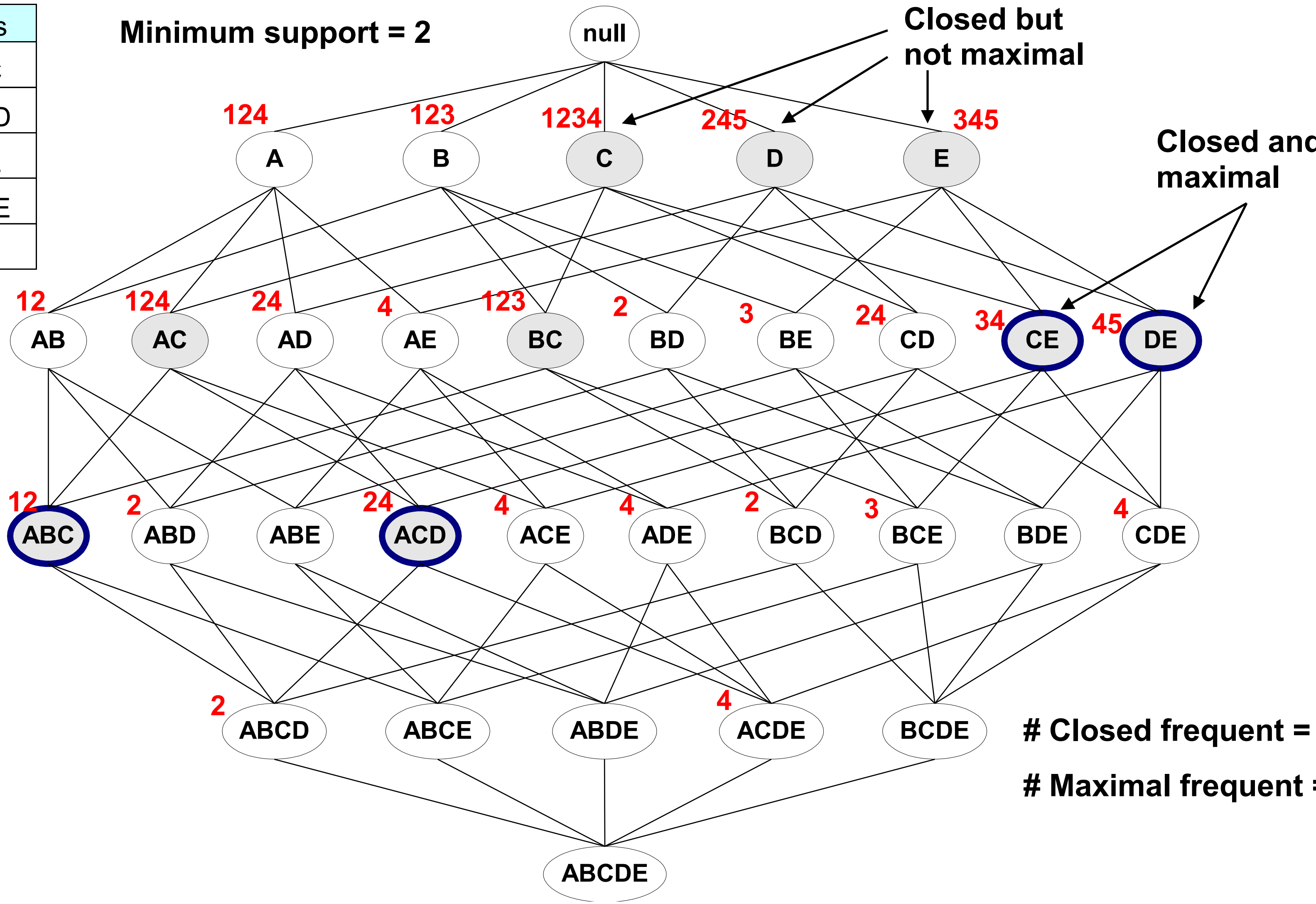
TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



Maximal Frequent vs Closed Frequent Itemsets

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE

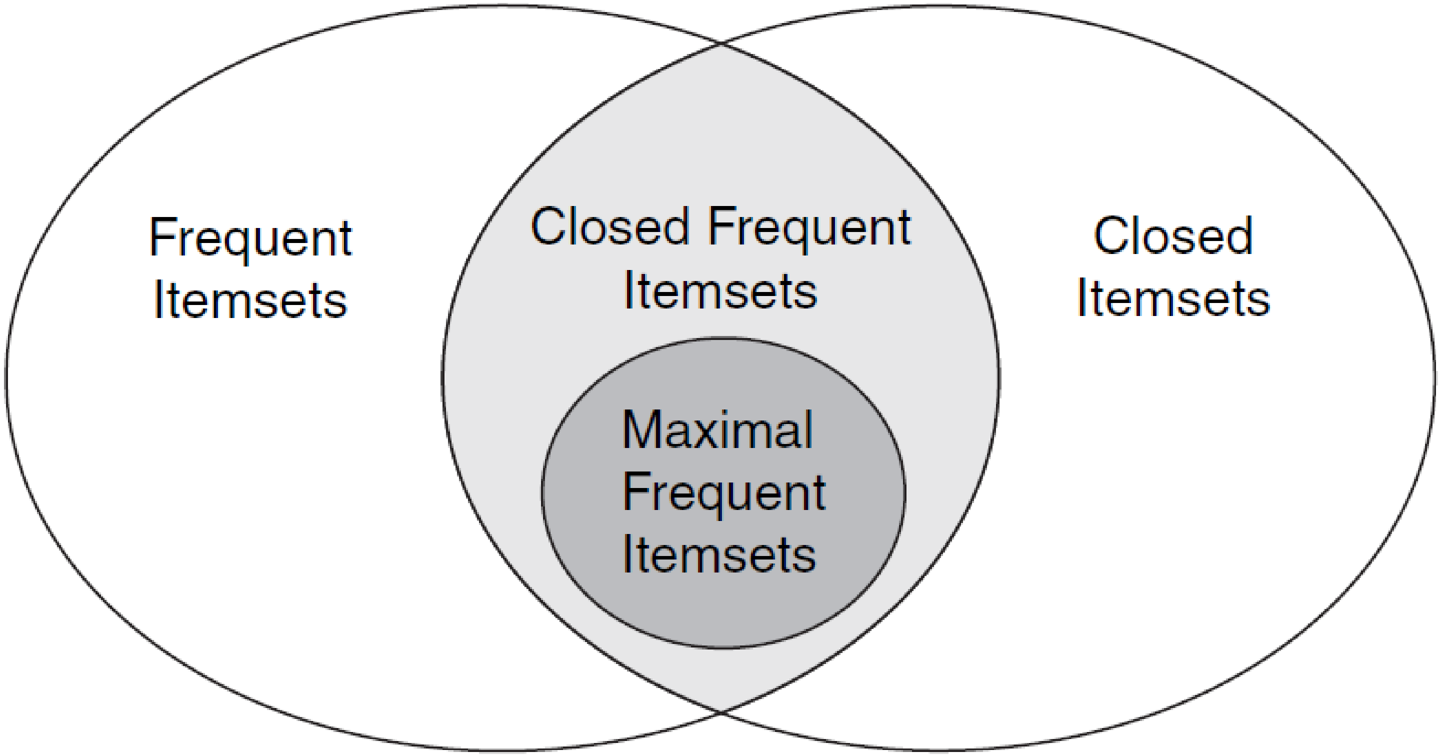
Minimum support = 2



 # Closed frequent = 9

 # Maximal frequent = 4

Maximal vs Closed Itemsets



Pattern Evaluation

- Association rule algorithms can produce large number of rules

- Interestingness measures can be used to prune/rank the patterns
  - In the original formulation, support & confidence are the only measures used

Computing Interestingness Measure

- Given $X \rightarrow Y$ or $\{X,Y\}$, information needed to compute interestingness can be obtained from a contingency table

Contingency table

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	N

f_{11}: support of X and Y
f_{10}: support of X and $\overline{Y}$
f_{01}: support of $\overline{X}$ and Y
f_{00}: support of $\overline{X}$ and $\overline{Y}$

Used to define various measures

- support, confidence, Gini, entropy, etc.

Drawback of Confidence

Customers	Tea	Coffee	…
C1	0	1	…
C2	1	0	…
C3	1	1	…
C4	1	0	…
…			

	$Coffee$	$\overline{Coffee}$	
Tea	150	50	200
$\overline{Tea}$	650	150	800
	800	200	1000

Association Rule: Tea $\rightarrow$ Coffee

Confidence $\cong$ P(Coffee|Tea) = 150/200 = 0.75

Confidence > 50%, meaning people who drink tea are more likely to drink coffee than not drink coffee

So rule seems reasonable

Drawback of Confidence

	Coffee	$\overline{\text{Coffee}}$	
Tea	150	50	200
$\overline{\text{Tea}}$	650	150	800
	800	200	1000

Association Rule: Tea $\rightarrow$ Coffee

Confidence= P(Coffee|Tea) = 150/200 = 0.75

but P(Coffee) = 0.8, which means knowing that a person drinks tea reduces the probability that the person drinks coffee!

$\Rightarrow$ Note that P(Coffee|$\overline{\text{Tea}}$) = 650/800 = 0.8125

Drawback of Confidence

Customers	Tea	Honey	…
C1	0	1	…
C2	1	0	…
C3	1	1	…
C4	1	0	…
…			

	$Honey$	$\overline{Honey}$	
Tea	100	100	200
$\overline{Tea}$	20	780	800
	120	880	1000

Association Rule: Tea $\rightarrow$ Honey

Confidence $\cong$ P(Honey|Tea) = 100/200 = 0.50

Confidence = 50%, which may mean that drinking tea has little influence whether honey is used or not

So rule seems uninteresting

But P(Honey) = 120/1000 = .12 (hence tea drinkers are far more likely to have honey

Measure for Association Rules

- So, what kind of rules do we really want?
  - $\text{Confidence}(X \rightarrow Y)$ should be sufficiently high
    - To ensure that people who buy X will more likely buy Y than not buy Y
  - $\text{Confidence}(X \rightarrow Y) > \text{support}(Y)$
    - Otherwise, rule will be misleading because having item X actually reduces the chance of having item Y in the same transaction
    - Is there any measure that capture this constraint?
      - Answer: Yes. There are many of them.

Statistical Relationship between X and Y

- The criterion

$$confidence(X \rightarrow Y) = support(Y)$$

  is equivalent to:

  - $P(Y|X) = P(Y)$
  - $P(X,Y) = P(X) \times P(Y)$ (X and Y are independent)

  If $P(X,Y) > P(X) \times P(Y)$: X & Y are positively correlated

  If $P(X,Y) < P(X) \times P(Y)$: X & Y are negatively correlated

Measures that take into account statistical dependence

$$Lift = \frac{P(Y \mid X)}{P(Y)}$$

$$Interest = \frac{P(X,Y)}{P(X)P(Y)}$$

lift is used for rules while interest is used for itemsets

$$PS = P(X,Y) - P(X)P(Y)$$

$$\phi - coefficient = \frac{P(X,Y) - P(X)P(Y)}{\sqrt{P(X)[1-P(X)]P(Y)[1-P(Y)]}}$$

Example: Lift/Interest

	Coffee	$\overline{\text{Coffee}}$	
Tea	150	50	200
$\overline{\text{Tea}}$	650	150	800
	800	200	1000

Association Rule: Tea $\rightarrow$ Coffee

Confidence= P(Coffee|Tea) = 0.75

but P(Coffee) = 0.8

$\Rightarrow$ Interest = 0.15 / (0.2×0.8) = 0.9375 (< 1, therefore is negatively associated)

So, is it enough to use confidence/Interest for pruning?

There are lots of measures proposed in the literature

Measure (Symbol)	Definition
Correlation (ϕ)	$\frac{N f_{11} - f_{1+} f_{+1}}{\sqrt{f_{1+} f_{+1} f_{0+} f_{+0}}}$
Odds ratio (α)	$\big(f_{11} f_{00}\big) / \big(f_{10} f_{01}\big)$
Kappa (κ)	$\frac{N f_{11} + N f_{00} - f_{1+} f_{+1} - f_{0+} f_{+0}}{N^2 - f_{1+} f_{+1} - f_{0+} f_{+0}}$
Interest (I)	$\big(N f_{11}\big) / \big(f_{1+} f_{+1}\big)$
Cosine (IS)	$\big(f_{11}\big) / \big(\sqrt{f_{1+} f_{+1}}\big)$
Piatetsky-Shapiro (PS)	$\frac{f_{11}}{N} - \frac{f_{1+} f_{+1}}{N^2}$
Collective strength (S)	$\frac{f_{11} + f_{00}}{f_{1+} f_{+1} + f_{0+} f_{+0}} \times \frac{N - f_{1+} f_{+1} - f_{0+} f_{+0}}{N - f_{11} - f_{00}}$
Jaccard (ζ)	$f_{11} / \big(f_{1+} + f_{+1} - f_{11}\big)$
All-confidence (h)	$\min\left[\frac{f_{11}}{f_{1+}}, \frac{f_{11}}{f_{+1}}\right]$

Simpson's Paradox

- Observed relationship in data may be influenced by the presence of other confounding factors (hidden variables)
  - Hidden variables may cause the observed relationship to disappear or reverse its direction!

- Proper stratification is needed to avoid generating spurious patterns

Simpson’s Paradox

- Recovery rate from Covid
  - Hospital A: 80%
  - Hospital B: 90%
- Which hospital is better?

Simpson's Paradox

- Recovery rate from Covid
  - Hospital A: 80%
  - Hospital B: 90%
- Which hospital is better?
- Covid recovery rate on older population
  - Hospital A: 50%
  - Hospital B: 30%
- Covid recovery rate on younger population
  - Hospital A: 99%
  - Hospital B: 98%

Simpson's Paradox

- Covid-19 death: (per 100,000 of population)
  - County A: 15
  - County B: 10
- Which state is managing the pandemic better?

Simpson's Paradox

- Covid-19 death: (per 100,000 of population)
  - County A: 15
  - County B: 10
- Which state is managing the pandemic better?
- Covid death rate on older population
  - County A: 20
  - County B: 40
- Covid death rate on younger population
  - County A: 2
  - County B: 5

Thank You

Slides Courtesy
1. Introduction to Data Mining, 2nd Edition
by Tan, Steinbach, Karpatne, Kumar
2. Prof. Carlos Castillo, UFB Barcelona